Project Planet: Efficient Display of Large Strategies

Michael Steele, Stuart Crawford
Fair, Isaac
August 15, 2000

I: Introduction

This document describes an R&D project currently underway in the Analytic Software Group. The effort is known as *Project Planet*, and is an effort to build a state-of-the-art tool for strategy display. The premise is that all strategy display tools currently in use at Fair Isaac (and elsewhere) are fundamentally flawed, and that a new approach to strategy display is required in order to remedy those flaws. In the sections below, more detail is provided regarding these flaws, and details of the solution provided by Project Planet are described. A separate document (entitled Strategy Designer Light) describes the first client of this new approach to strategy display.

II: What s wrong with the current tools?

The current crop[1] of strategy display tools has a variety of problems, as follows:

- **They are all different!** Even though these tools all share the same fundamental goal (to allow for the creation, viewing, and editing of strategies) they each are implemented using a completely separate set of code, and are based upon different GUI designs.

- **They do a very poor job of displaying large strategies.** Historically, strategies at Fair, Isaac have been small. This is certainly true, in large part, because the strategies were constructed manually. One could also argue that the strategies were kept *artificially* small due to the fact that large strategies are so difficult to view and interact with using the existing set of tools. With the advent of Workbench Strategy Designer (WSD) and Prevision, strategies are *automatically* constructed and tend to be large and sometimes (as is the case with Prevision) *very* large. Even though the strategies are automatically constructed, it is still critical to be able to view and edit them. Even the strategy-editing tool of WSD (which is the most sophisticated of the current crop of tools) does a pretty atrocious job of displaying such large strategies.

All these tools employ the same layout technique: enough distance to fit all their descendant nodes between them separates the branches of a split. So when displaying a large strategy, the branches of the splits closest to the root become too separated to see at

[1] TRIAD Treeview, Workbench Strategy Designer, StrategyWare, Decision System

once. This makes it difficult to edit and understand the splits at the very top of the strategy, which are the most important splits in the strategy.

- **Navigation through the strategy is inefficient**. The traditional tools of scroll and zoom are poor approaches to strategy navigation. It is all too easy to lose one s place within the strategy and to lose context. These tools require the user to know the place on the canvas to which they want navigate. When the tree becomes even moderately sized, it becomes difficult to remember where on the canvas a certain branch is located. It can require much trial and error before the desired branch is found.

- **The condition path is difficult to follow**. The sequence of conditions that must be met in order to reach any segment in the strategy is the *condition path*. The condition path is a critical element of the strategy because it answers the question How do I get here? Despite this critical need, none of the current tools provides a way to easily view the condition path

III: How does Planet address these problems?

The approach embodied within Planet draws from the design principles of the new field of Information Visualization, advocated by Tufte (The Visual Display of Quantitative Information) and specifically makes use of display techniques such as fish-eye views (Furnas, Lamping, Rao, Pirolli). Details of how these principles address the problems described in the previous section follow below.

Design Principle 1: Don t let the bushiness of the offscreen part of strategy affect the display of the onscreen part of the strategy

The traditional approach to displaying a large strategy is fundamentally flawed because the *entire* strategy is first laid out in memory, even though only part of that strategy is viewable at any time. This leads to phenomena where the most important part of the strategy (the initial branches) cannot be viewed because the offscreen portion of the strategy is so bushy. The best one can hope to do is get a sense of those initial branches by scrolling and zooming, an approach that is, at best, inefficient. The approach taken by Planet is very different: if a portion of the strategy is not being viewed, it has no effect on the layout of the visible portion of the strategy. This means that the strategy layout is dynamic and adaptable to the current portion of the strategy being viewed. It is not static as is the case of the current set of tools. This allows the user to view, in its entirety, the portion of the strategy on which they currently want to concentrate.

Design Principle 2: Don t try to show detail where it isn t important.

The traditional approach to strategy display shows every segment in the strategy with the same amount of detail. If you zoom in, every segment gets bigger and if you zoom out every segment gets smaller. Contrast this with the fisheye approach (Furnas) that defines a single segment as the focal point of the display and displays segments with less detail the farther away they are from the focal point. This makes for a very efficient use of screen real estate and dispenses with the need to provide a traditional zoom capability.

Design Principle 3: Always show the condition path.

As described above, the condition path is a critical piece of information to the viewer of a strategy. Planet always displays the set of conditions needed to reach the single segment currently selected as the focal point.

Design Principle 4: No scroll bars.

Scroll bars are fine for scrolling around an image, where the width of an atomic piece of information is a pixel. In the context of a strategy, the atomic piece of information is a segment, and providing scrollbars that allow you to scroll a pixel at a time is just silly. Instead of scroll bars, navigation takes place by hopping from segment to segment.

Design Principle 5: Provide Navigational Shortcuts

In the traditional tools, the only way to navigate to a particular point in the strategy is via scrolling. In Planet, a double-click on any segment makes that the focal point and a double-click on any element in the decision path makes the corresponding segment the focal point.

Design Principle 6: Provide Navigational Cues

Because the gradual scroll provided by the scroll bar is not present, it is necessary to use another way to provide a gentle transition when the focal point is changed. Smooth, double-buffered animation transitions are used for this purpose.

Design Principle 7: Fit as much information on the screen as possible

Planet is designed to fit as much information into a single eye gulp as possible by making efficient use of screen real estate. First, available space is utilized to provide extra context for the focus node. For example, the siblings of the focus node are displayed in the empty area to the left of the focus node s descendants. Second, redundant information

is eliminated. For example, when all the branches of a split are conditioned on the same variable, the name of the variable is printed just once, in the split node, rather than multiple times, once on each branch. Third, information is rendered as compactly as possible. The widths of nodes and levels are just wide enough to fit the widest label.

Design Principle 8: Maintain a consistent root-children orientation

Some approaches to tree visualization (Lamping, et al. 1995) allow the root of the tree to be placed dynamically *anywhere* in the display. Our view is that this approach, while useful from a purely quantitative display point-of-view, is not useful to the strategy consultant, who is used to viewing strategies that have (for example) the root placed in a predictable, consistent location. As such, we have constrained the planet treeviewer to adhere to this constraint: the root is always at the center, leftmost portion of the display.

Design Principle 9: Fit the tree display into a rectangular view

Some approaches to tree visualization (Lamping, et al. 1995) require that the tree be rendered in a circular view. Computer screens are rectangular, and such an approach results in wasted screen real estate. As such, we render our display so that it makes most efficient use of a rectangular view.

Design Principle 10: Be capable of rendering all Fair Isaac strategies.

Some of the existing strategy display tools are designed to render only certain, limited, kinds of strategies. For example, the TRIAD treeviewer can only render strategies that use the same split variable in each split on a level. Planet makes no assumptions about the form of the strategies that it is rendering. Every strategy that Fair Isaac creates can be viewed with Planet.

Illustrations:

The two figures that follow compare the display of a portion of a strategy in Strategy Designer (an example of a conventional tree viewer) and in Planet. The strategy is too large to be displayed in full with either tree viewer. The figures are the same size and use the same font size for node labels so as to present as fair of a comparison as possible. The portion of the strategy shown in both displays begins with a split on MonthsOnBooks .

Here are some differences:

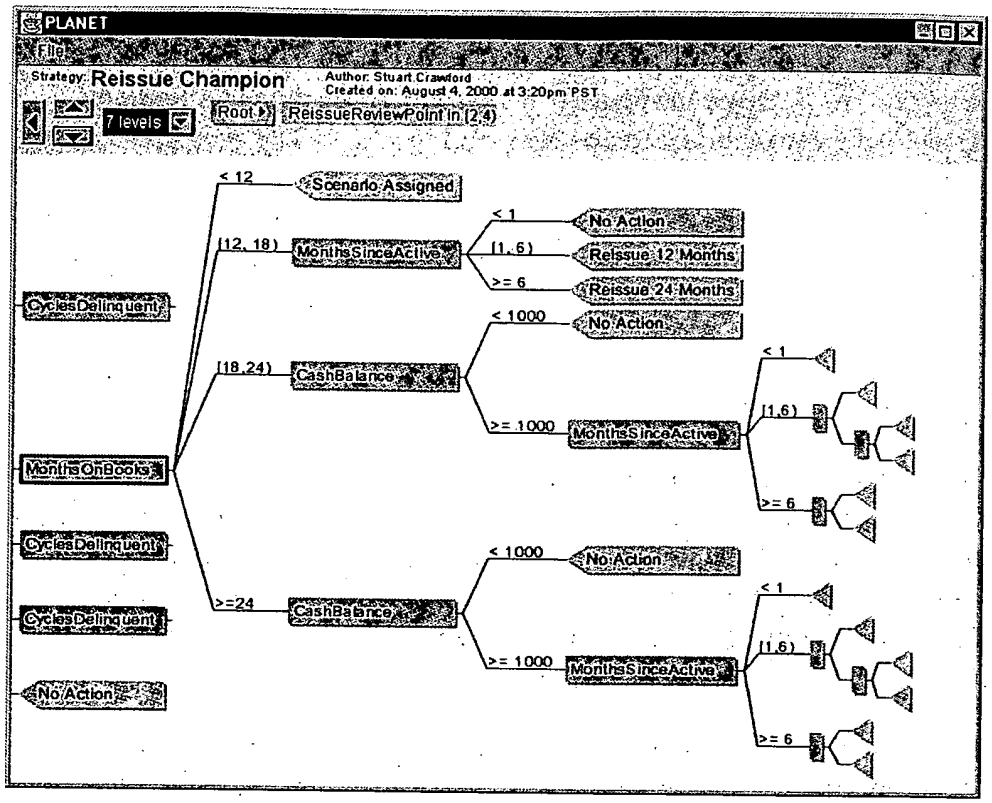
1.) The only indication given by Strategy Designer of our current location in the strategy is the position of the scroll bars, which indicate that we are somewhere in the middle of the tree. Planet not only indicates that we are one level from the root, it also indicates that the condition that leads us to our current position is ReissueReviewPoint in [2,4) (meaning that ReissueReviewPoint must be between 2 and 4 to get to this branch).

2.) In Strategy Designer, you cannot see the MonthsOnBooks split in its entirety. The MonthsOnBooks >= 24 condition is scrolled below the window. With large trees, this is quite a problem: often you cannot see more than one condition of a split at any time. If you try to zoom out enough to see all the conditions, the condition labels become too small to read. In Planet, this is never a problem. The leftmost splits are always displayed in full as shown in this example.

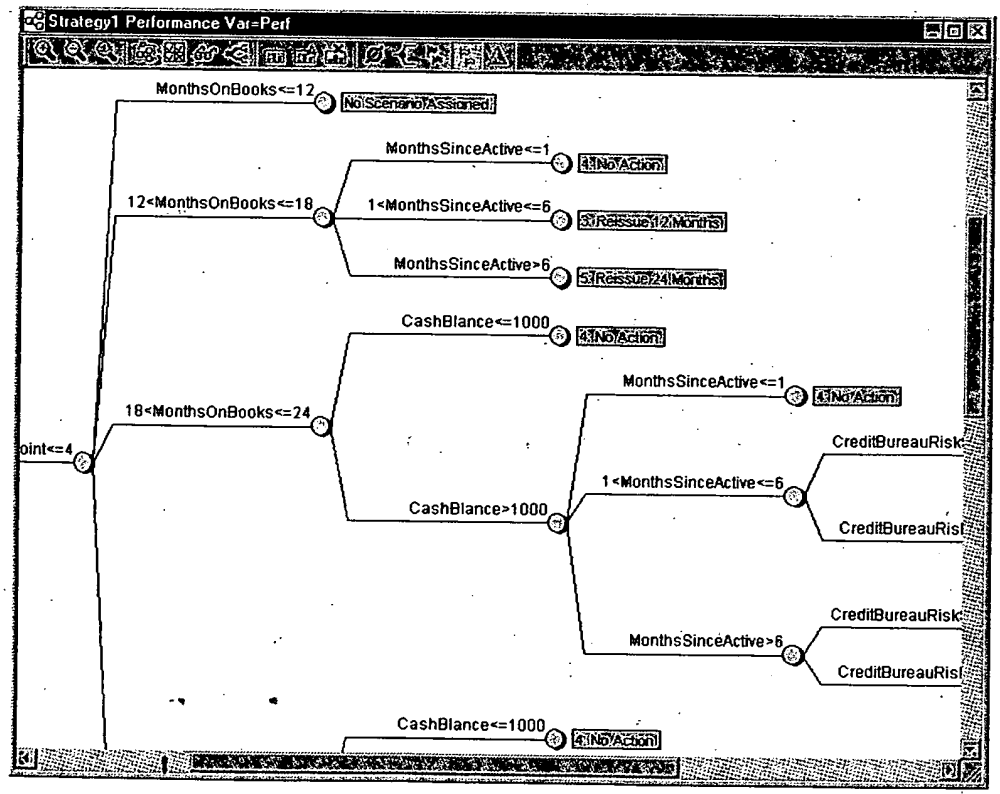
3.) In Strategy Designer, you cannot see the siblings of the MonthsOnBooks split. In Planet, the siblings are displayed above and below the MonthsOnBooks split . This makes for easier comparing, contrasting, and navigation between siblings.

4.) In Strategy Designer, there is no clue as to what the branch looks like off the right edge of the window. You cannot tell how many levels deep it is or how many leaves it has. In Planet, a shrunken version of this portion of the branch is displayed to present this valuable contextual information.

Let s say you want to navigate to the parent node of the MonthsOnBooks split so you can edit it. In Planet, you simply click the left arrow button to hop to it . If you want to navigate to a sibling node, you click the up arrow button or the down arrow button. Navigation is simple because you can hop from node to node. In Strategy Designer, you must use the scroll bars to reposition your view of the canvas. This means you have to locate (or remember) where on the canvas the node you are looking for is displayed. For large trees, this is like finding a needle in a haystack.



The Planet Tree Viewer



The Strategy Designer Tree Viewer

IV: Implementation

Planet is implemented in Java. The choice of Java as a development language was motivated as follows:

- **Speed**: We wanted to develop Planet *quickly*. Industry studies indicate that programmer productivity is enhanced by at least a factor of five over languages like C++. This was borne out in our experience with Planet. The richness of the Java platform, such as automatic double-buffering for animations and automatic antialiasing for smoother-looking graphics, made it possible to very rapidly build the software. Planet, in its current (early August) state, was implemented by a single developer (a complete newcomer to Java) working half time for less than two months. I don t believe that even a deeply experienced C++ developer could deliver the same level of functionality in so robust a package in such a timeframe.

- **Small Footprint**: We wanted Planet to be small. We expect that the finished code will be no larger that 100Kb. There is no way similar functionality could be provided in such a small package using (say) C++.

- **Portability**: We wanted to be platform and browser neutral. Java is the natural choice.

- **Connectivity**: We wanted easy connectivity to servers. Once again, the built-in Java networking classes made this development straightforward.

Now, it would certainly be possible to develop Planet in C++. However, the resulting code would be larger, more difficult to implement and (potentially) less robust. Planet has a small memory footprint and garbage collection is unlikely to pose a problem. Even if it did, the problem could be solved in a straightforward way via a simple object-pooling scheme. Planet is an extremely poor choice for delivery via a Windows Terminal Server mechanism. The animation that is used to provide the critical navigational cues would display poorly in such an environment. Planet needs to run on the client machine. Fortunately, the small size of Planet makes this a trivial issue.

V: Status

Design of the Planet strategy display GUI began in mid-April. Development of the Planet code began in earnest in mid-June, and an alpha (functionally complete) version will be available by the beginning of September.